INTERIM REPORT

on a

CODING SYSTEM DESIGN FOR ADVANCED SOLAR MISSIONS

Contract NAS2-3637

Submitted to:

AMES RESEARCH CENTER

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

October 20, 1966

Submitted by:

CODEX CORPORATION
222 Arsenal Street
Watertown, Mass. 02172

Table of Contents

Introduction

This Interim Report on the work done during the first part of Contract NAS2-3637, "Coding System Design for Advanced Solar Missions", presents the results of various studies performed by Codex Corporation in close cooperation with personnel of Ames Research Center. In the following paragraphs we summarize the work which has been done to date, and then present our recommendations for the studies to be completed during the remainder of the program.

The report is organized in the form of a brief report of results and recommendations, with all details relegated to a series of appendices presenting the necessary supporting data.

Summary of Work Performed

The first phase of this contract was intended to narrow down the choice of codes for the extended Pioneer mission, and to specify the parameters for some likely schemes in full detail.

To this end, extensive computer simulations of different coding schemes have been performed. We have written programs to simulate orthogonal equation decoding (OED) of both convolutional and block codes on a white gaussian channel; these programs have then been run at Ames

to simulate a variety of schemes. A simulated sequential decoder has been programmed in FORTRAN at Ames, and used for extensive simulations. The results of all these simulations are discussed in detail in Appendix A. Finally, we have written and run a short computational program to evaluate different equal-step quantization schemes, which we discuss in Appendix B.

These simulations clearly established the performance superiority of sequential decoding over the orthogonal equation decoding schemes. We therefore moved quickly to determine whether a sequential decoding scheme would be practical. The initial approach we took, after close consultation with Ames, was to consider the encoder as an outrigger box, like an experiment, receiving the output of the DTU and generating a signal for the RF modulator; preliminary design is now underway, under the guidelines reported in Appendix D. For decoding, we first investigated whether the on-site general purpose SDS 910 or 920 computers could be used for the bulk of decoding in real time with the 7094 at Ames as a backup. Ames is now looking closely at the interfaces which will be required on-site and at the Tape Processing Station at Ames; meanwhile, we have written machine language sequential decoders for the 910-920 and for the 7094. In the latter are included some optional tabulation routines which make the program suitable for further simulations. The characteristics of these programs are discussed in Appendix C. Finally, as a backup, we have begun to think about the implementation of a special-purpose sequential decoder.

In the next section, using the results of investigations to date, we summarize our current thinking on which scheme should be chosen and how to implement it.

In all the work to date, we have greatly appreciated being able to talk frequently with Dr. Lumb and his group at Ames; this close consultation has been the principal reason for the rapidity of our progress to date. Particularly helpful were various visits, of Codex' F. Peltz and D. Forney to Ames, and of Ames' L. Hofman and Dr. Lumb to Codex.

Recommendations

The simulations described in Appendix A confirm the expected significant difference in performance between the OED schemes and sequential decoding. Under the most optimistic assumptions, the most powerful convolutional code with OED buys 2 1/2 db over the current system, while the most powerful block code might barely attain 3 db. In contrast, the sequential decoding system delivers almost completely clean data at a signal-to-noise ratio 4 db lower than the current system, and data with an error rate equivalent to that of the current system at 5 db improvement. Its performance advantage over the OED systems is quite significant both in quantity and quality of data returned for a given power.

The studies of implementation we have made to date have not raised any doubts on the feasibility of sequential decoding, particularly in comparison to the OED schemes. The 24-stage convolutional encoder required on board is almost identical to that in the convolutional OED scheme and considerably simpler than that in the block OED scheme. There seems a good chance that the ground decoding can be done in already existing general-purpose computers, at the cost only of some additional interfacing;

the OED schemes can do no better, except perhaps in time and storage requirements. If we are forced to special-purpose decoding, the decoder required for full OED ought to be comparable to a special-purpose sequential decoder in complexity, though we have not yet made the detailed studies to justify this.

In view, therefore, of the performance advantage and the lack of identified implementational disadvantages of sequential decoding vis-a-vis OED decoding, we feel that no further effort on the latter ought to be undertaken unless unforeseen drawbacks in sequential decoding appear. Further effort ought to be concentrated on developing as completely as possible a system using sequential decoding of a rate-1/2 convolutional code. In this work some significant tasks are:

1) Determine an effective criterion for detecting errors made by the sequential decoder.

2) With this criterion and with 1-3-5-7 parity, re-evaluate a set of codes and settle on the best. Codes up to constraint length 25 ought to be considered, in view of the fact that the SDS programs can conveniently handle these constraint lengths.

3) With this code, determine the optimal decoding program, including choices of algorithm, bias, and threshold spacing.

4) Through simulating the entire system as completely as possible, check that no significant degradations from performance with white gaussian noise occur. If they do, find a fix if possible.

5) Construct and evaluate a fast, efficient synchronization program for initial synchronization and resynchronization. Establish a useful criterion for detecting loss of synch.

6) Carry forward the detailed design of the spacecraft encoder as far as possible.

7) Examine the implementation of the decoder in specially furnished hardware, including the possibility of a programmed small general-purpose computer.

Appendix A. Results of Simulations

Three types of coding schemes were simulated on a white gaussian channel: convolutional encoding with orthogonal equation decoding (OED), block coding with OED decoding, and convolutional encoding with sequential decoding. We shall describe each group of simulations in turn, but first we briefly discuss the general features of binary communication over a gaussian channel.

1. General Considerations

With binary signaling on a white gaussian channel, performance is characterized entirely by the signal-to-noise ratio per information bit, E_b/N_o. For a rate-1/2 code, $E_b/N_o = 2\beta$, where β is the signal-to-noise ratio per transmitted bit, equal to the ratio of the energy E in a single transmitted bit to the single-sided spectral density of the noise, N_o. In general, for a code of rate R, $E_b/N_o = \beta/R$. It is well known that, under a convenient normalization, the output of a perfectly coherent demodulator is a gaussian random variable of mean $\pm 2\beta$, depending on whether a zero or one was sent, and variance 2β. In the simulations we assume that the all-zero sequence is sent, which in no case affects performance, and assume that the demodulator output is quantized to 2^m levels, which we call m-bit quantization. In Appendix B we see that performance with 6-bit, equal-step quantization is essentially identical to that obtainable with arbitrarily fine quantization; we use such quantization with the OED schemes to find the limits of their performance. With sequential decoding, 3-bit quantization is used, as would be the limit in practice; Appendix B shows that the resulting degradation is very small.

We also desired to take into account the redundancy in the Pioneer data format. In general, we went under the assumption that every seventh bit in the information stream would be an odd parity check on the preceding 6. This added information allowed in the OED decoding several kinds of strategies of erasure of 7-bit words, whose purpose was to reduce the undetected error probability at the expense of some erasure probability; in sequential decoding it permitted an efficient forcing of the search. We have lately been informed that actually the seventh bit in the Pioneer format is an odd parity check on the bits two, four, and six bits before, but of course this bit could be replaced in the encoder by the overall parity check which is assumed here.

2. Convolutional Codes with OED Decoding

Orthogonal equation decoding is a method of using a number of equations, each including the bit to be decoded, but including no other bit more than once, to compute a reliable log likelihood ratio of the decoded bit from the received log likelihood ratios of the bits in the equations. The method has been discussed in detail in other reports, and bounds derived on its performance. (1-2) We noted in particular that whereas in previously suggested schemes bits already decoded were assumed correct (hard decision), a performance improvement might be obtained by assigning these bits the log likelihood ratios computed in the decoding process (full likelihood). We were therefore concerned with the performance achievable at low signal-to-noise ratios by both these schemes, and with a comparison between them. In all cases, therefore, the same noise sequence was used for both.

We also wished to determine how best to use the seventh bit overall parity check. An obvious strategy is to reject any words in which parity does not check, and accept the rest. However, we felt that we might do better to use the

magnitudes of the computed log likelihood ratios, as well as their signs, to get a more sensitive test, as had previously been suggested at Ames (3). Here we simply computed the probability that the entire word was correct, given the computed log likelihood ratios and assuming independence between bits, which of course is false. Any word with probability less than .99 of being correct was rejected; naturally any word whose parity did not check was rejected. A third strategy we tried was the following: in words in which parity failed to check, we first changed the least likely bit in the block, and then applied the test described above on the word including the changed bit.

It is characteristic of most decoding schemes for convolutional codes that when one decoding error is made, several others tend to follow before the decoder becomes resynchronized. We call such an event an error run, and define it precisely as follows:

1.) A bit is the last bit of an error run if it is decoded incorrectly and the succeeding n-1 bits are decoded correctly, where n is the decoding constraint length;

2.) A bit is the first bit of an error run if it is decoded incorrectly and the decoder is not already in an error run.

With this definition, we can then define the first error probability P_1 as the probability of entering an error run, given that the decoder is not already in a run. The overall bit error probability P_o is then approximately equal to $\bar{e}P_1$, where $\bar{e}$ is the average number of errors in a run. We also tabulate the average length of a run, $\bar{\ell}$, and the average error density in runs, $\bar{e}/\bar{\ell}$.

In the end, only two codes were simulated, the (44, 22) and (24,12) trial-and-error codes of Massey (4), at the three values of E_b/N_o of 2.4 (3.8 db), 3.6 (5.5 db), and 4.7 (6.8 db), and with both hard decision (HD) and full likelihood (FL) feed-back. The pertinent results are tabulated in Table I. Besides the total

Table I

Performance of (24,12) and (44,22) codes with OED-HD and OED-FL decoding.

Code	Decoding Method	E_b/N_o (db)	N	P_o	P_1	$\bar{e}$	$\bar{\ell}$	$\bar{e}/\bar{\ell}$	P_{x1}	P_{e1}	P_{x2}	P_{e2}
(24,12)	OED-HD	6.8	100,000	4.0×10^{-5}	1.0×10^{-5}	4.0	10	.4	1.4×10^{-4}	0	1.4×10^{-4}	0
(24,22)	OED-HD	5.5	50,000	1.8×10^{-3}	2.0×10^{-4}	9.0	26	.34	3.8×10^{-3}	2.1×10^{-3}	5.0×10^{-3}	1.3×10^{-3}
(24,12)	OED-HD	3.8	50,000	2.9×10^{-2}	3.5×10^{-3}	8.2	27	.33	6.5×10^{-2}	4.0×10^{-2}	10.4×10^{-2}	1.35×10^{-2}
(24,12)	OED-FL	6.8	100,000	1.0×10^{-5}	1.0×10^{-5}	1	1	1	7.0×10^{-5}	0	7.0×10^{-5}	0
(24,12)	OED-FL	5.5	50,000	5.2×10^{-4}	2.2×10^{-4}	2.5	7.2	.33	2.5×10^{-3}	1.4×10^{-4}	3.4×10^{-3}	1.4×10^{-4}
(24,12)	OED-FL	3.8	50,000	9.0×10^{-3}	4.4×10^{-3}	2.1	6.4	.33	4.0×10^{-2}	9.2×10^{-3}	7.9×10^{-2}	5.5×10^{-3}
(44,22)	OED-HD	6.8	100,000	0								
(44,22)	OED-HD	5.5	50,000	1.2×10^{-4}	4.0×10^{-5}	3.0	8.0	.38	2.8×10^{-4}	2.8×10^{-4}	2.8×10^{-4}	2.8×10^{-4}
(44,22)	OED-HD	3.8	50,000	6.9×10^{-3}	7.4×10^{-4}	9.3	37	.26	1.75×10^{-2}	9.5×10^{-3}	2.6×10^{-2}	4.8×10^{-3}
(44,22)	OED-FL	6.8	100,000	0								
(44,22)	OED-FL	5.5	50,000	4.0×10^{-5}	4.0×10^{-5}	1	1	1	2.8×10^{-4}	0	2.8×10^{-4}	0
(44,22)	OED-FL	3.8	50,000	2.8×10^{-3}	1.0×10^{-3}	2.8	12	.23	1.2×10^{-2}	2.9×10^{-3}	2.1×10^{-2}	9.8×10^{-4}

number of information bits decoded, N, and P_o, P_1, $\bar{e}$, $\bar{\ell}$, and $\bar{e}/\bar{\ell}$, defined in the previous paragraph, we tabulate the word erasure probabilities, P_{x1} and P_{x2}, and the word error probabilities, P_{e1} and P_{e2}, using the first two schemes described in the paragraph preceding the last. The performance of the third erasure scheme was in every case nearly identical to that of the second.

Finally, we have plotted in Figure 1 the first error probability against E_b/N_o, for both codes, with OED-HD, in order to compare these results with those computed earlier (1).

DISCUSSION

Our principal interest, of course, is in performance. Let us therefore examine closely the result obtained with the most powerful code (44,22), decoding scheme (OED-FL), and word erasure strategy (the second). At an E_b/N_o of 3.8 db, which is exactly 3 db below the E_b/N_o at which with PSK the error probability is 10^{-3}, this scheme has a bit error probability of 2.8×10^{-3}, or greater than the target probability of 10^{-3}.

More importantly, with the erasure strategy previously described, the word erasure probability is about 2%, and the word error probability, about 10^{-3}; this is to be compared with the erasure probability of .7% and error probability of 2×10^{-5} which would be obtained with PSK at a bit error rate of 10^{-3}, with the simplest erasure strategy. It is clear, therefore, that the performance of this scheme at $E_b/N_o = 3.8$ db is inferior to the standard set by the current system.

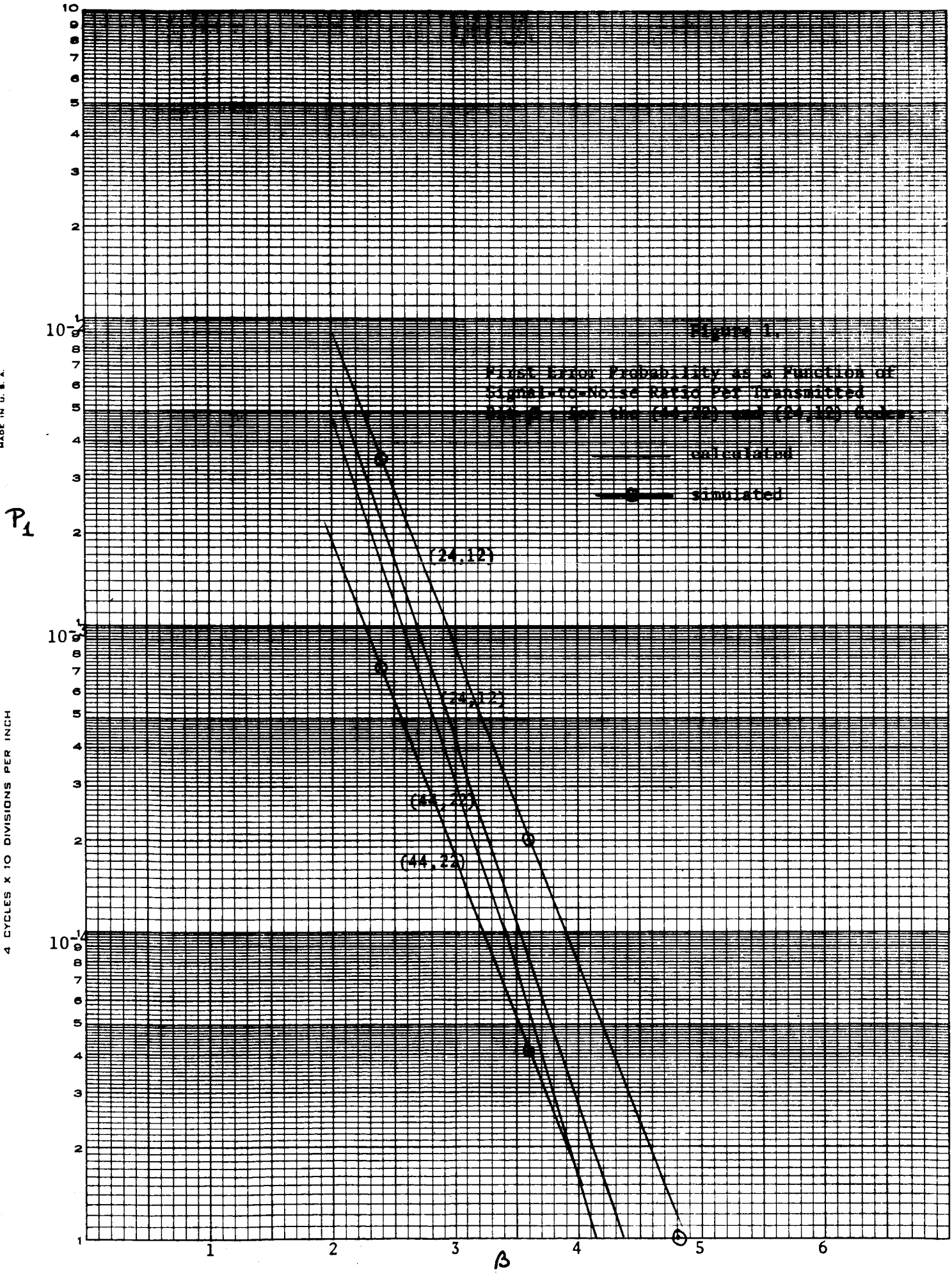


Figure 1.

First Error Probability as a Function of Signal-to-Noise Ratio Per Transmitted Bit (β) for the (44,22) and (24,12) Codes.

That at 5.5 db is clearly better, however; probably the two are about the same at 4.4 - 4.5 db. Since we want no longer code (the next in this sequence is (104,52), and know no more powerful decoding scheme for these codes, we see that with convolutional codes and OED decoding we can gain no more than about 2 1/2 db over straight PSK, for a target bit error rate of 10^{-3}. This is our major conclusion; while somewhat more heartening than those reported before, due principally to the introduction of full likelihood feedback, it is hardly earth-(or heaven-) shaking.

There are some interesting detailed comparisons that can be made, however. First, let us compare hard decision with full likelihood feedback. We see that the first error probabilities for each hardly differ, and that in fact the hard decision results are superior, which is to be expected since as long as no error has been made, feeding back infinite likelihoods is all to the good. However, the average number of errors in a run is markedly less for full likelihood than for hard decision feedback: typically 2 or 3 in the former case, 8 or 9 in the latter. Evidentally attaching a low likelihood to an error is very effective in cutting down propagation. There results an improvement in bit error probability of a factor of 3 or 4, which is equivalent to a gain of between 1/2 and 1 db for the error probability-E_b/N_o slopes typical of these codes. When we examine the results of the word erasure strategies, moreover, we see that for some reason they are noticeably more effective with full likelihood feedback than with hard decisions, again undoubtedly a manifestation of FL's greater sensitivity to errors resulting in more equivocal log likelihood ratios. Thus we can say that full likelihood feedback buys roughly 1 db over hard decision; since the implementation for both is nearly identical, it looks as though the former is preferable.

As for the erasure strategies, we see that both are basically handicapped by having to work against highly clustered errors. This results in general in a somewhat lower erasure rate than a BSC would exhibit for the same bit error rate, but a much higher error rate; these two quantities may be traded off against one another by varying the threshold in the second strategy, but the net result is inferior to the BSC. Some sort of scrambling or interleaving is therefore suggested. In comparing the strategies to one another, we see that with a threshold of .99 we tend to increase the word erasure probability by less than a factor of two, while decreasing the error probability by a factor slightly greater than 2. Undoubtedly, a still higher threshold would be preferable. The reason for the negligible difference between the second strategy and the third strategy, in which the least likely bit is flipped before computation of the word likelihood, is that it is extremely unlikely that, after flipping, the word likelihood will be very high, so that practically all words whose parity does not check are rejected anyway.

Finally we note the curious fact that in the (24,12) code, which is basically three-error-correcting, the error density in runs is very nearly 1/3, while in the 4-error-correcting (44,22) code, it is 1/4. Professor James Massey reports that this phenomenon has been repeatedly observed but never explained.

3. Block Codes With OED Decoding

For certain block codes, a set of orthogonal parity check equations can be formed, and thus OED decoding can be used. Among these codes one of the simplest is the (15,7), a distance-5 code, while one of the more complicated is the (73,45), a code of distance 10; these were the two codes simulated. As with convolutional codes, both hard decision and full likelihood feedback are possible.

The efficient way to use the seventh bit parity check in a block code is not clear. First, when there is an error in a block, the whole block tends to have low decoded log likelihood ratios, and should probably be thrown out as a unit. In this decision, the seventh parity bit can help, but it is neither natural nor efficient. Since a block code inherently requires considerable buffering anyway, we now feel that the most effective solution would be to take out the overall parity bit in the encoder, and to use a rate 6/14 block code so as to keep the transmitted bit rate twice the incoming bit rate. Codes of rate 6/14 can be obtained by shortening longer codes; for example, the (15,7) can be shortened to (14,6), or the (73,45) to (49,21). The alternative would be to take out the overall parity bit and use the more powerful unshortened block code; then the transmitted bit rate would no longer be simply related to the information bit rate, and considerable modification of encoder and demodulator timing would be required.

Table II summarizes the performance obtained with the shortened (14,6) and (49,21) codes, with orthogonal equation decoding and full likelihood feedback. The signal-to-noise ratios per transmitted bit considered were 0.3 and 0.8 db, corresponding to signal-to-noise ratios per original information bit (before excising the seventh bit) of 3.3 and 3.8 db. In this table N is the total number of words decoded, P_{ew} the percentage of words in error, and P_{eb} the fraction of bits in error.

Table II

CODE	DECODING METHOD	E_b/N_o (db)	N	P_{ew}	P_{eb}
(14,6)	OED-FL	3.8	30,000	1.7×10^{-2}	3.0×10^{-3}
(14,6)	OED-FL	3.3	30,000	2.8×10^{-2}	5.4×10^{-3}
(49,21)	OED-FL	3.8	2,000	2.5×10^{-2}	1.0×10^{-3}
(49,21)	OED-FL	3.3	2,000	6.5×10^{-2}	3.8×10^{-3}

In Table III, we tabulate the performance obtained with the unshortened (15,7) and (73,45) codes. Here E_b/N_o has been adjusted as above to reflect the gain in dropping the seventh parity bit. We have also tabulated P_{ei} and P_{ep}, the probability of error in the information bits and in the parity bits, where the first 7 or 45 bits in a word are considered the information bits.

Table III

CODE	DECODING METHOD	E_b/N_o (db)	N	P_{ew}	P_{eb}	P_{ei}	P_{ep}
(15,7)	OED-HD	6.4	30,000	6.7×10^{-5}	2.2×10^{-5}	1.4×10^{-5}	2.9×10^{-5}
(15,7)	OED-HD	5.1	17,232	7.6×10^{-4}	2.6×10^{-4}	2.1×10^{-4}	3.0×10^{-4}
(15,7)	OED-HD	5.1	30,000	8.7×10^{-4}	3.2×10^{-4}	2.4×10^{-4}	3.9×10^{-4}
(15,7)	OED-HD	3.4	30,000	1.7×10^{-2}	6.4×10^{-3}	4.8×10^{-3}	7.7×10^{-3}
(15,7)	OED-HD	3.4	30,000	1.7×10^{-2}	6.3×10^{-3}	5.1×10^{-3}	7.5×10^{-3}
(15,7)	OED-FL	6.4	30,000	6.7×10^{-5}	4.4×10^{-6}	9.5×10^{-6}	----
(15,7)	OED-FL	5.1	17,232	9.9×10^{-4}	1.9×10^{-4}	2.0×10^{-4}	1.7×10^{-4}
(15,7)	OED-FL	5.1	30,000	1.3×10^{-3}	1.8×10^{-4}	2.2×10^{-4}	1.4×10^{-4}
(15,7)	OED-FL	3.4	17,000	2.2×10^{-2}	3.8×10^{-3}	4.3×10^{-3}	3.4×10^{-3}
(15,7)	OED-FL	3.4	30,000	2.2×10^{-2}	3.8×10^{-3}	4.4×10^{-3}	3.3×10^{-3}
(73,45)	OED-HD	5.2	5,000	-----	-----	-----	-----
(73,45)	OED-HD	3.9	5,000	6.2×10^{-3}	1.1×10^{-3}	2.3×10^{-4}	2.4×10^{-3}
(73,45)	OED-HD	2.2	2,500	1.6×10^{-1}	3.3×10^{-2}	1.3×10^{-2}	6.7×10^{-2}
(73,45)	OED-FL	5.2	5,000	-----	-----	-----	-----
(73,45)	OED-FL	3.9	5,000	6.8×10^{-3}	1.7×10^{-4}	2.3×10^{-4}	7.8×10^{-5}
(73,45)	OED-FL	2.2	2,500	1.9×10^{-1}	8.4×10^{-3}	1.0×10^{-2}	5.7×10^{-3}
(73,45)	OED-FL	2.2	2,000	1.8×10^{-1}	8.0×10^{-3}	9.3×10^{-3}	6.1×10^{-3}
(73,45)	OED-FL	1.7	2,000	3.1×10^{-1}	1.6×10^{-2}	1.8×10^{-2}	1.2×10^{-2}

Finally, in Figure 2 we have plotted the bit probability of error P_{eb} against E_b/N_o (not in db) for the (73,45) code with full likelihood feedback.

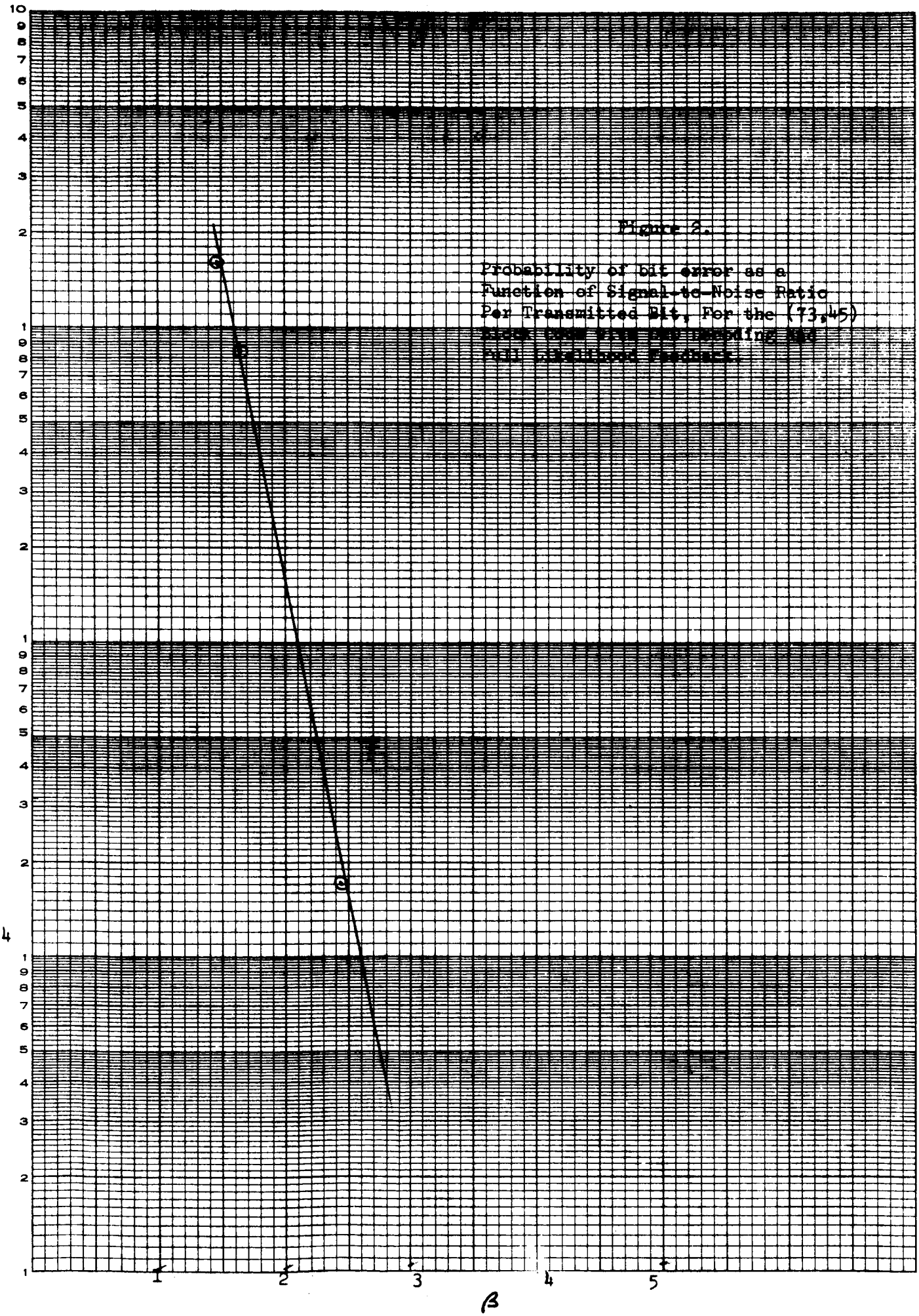


Figure 2.

Probability of bit error as a Function of Signal-to-Noise Ratio Per Transmitted Bit, For the (73,45) Block Code With Iterative Decoding And Full Likelihood Feedback.

DISCUSSION

Again, let us first consider the best performance obtainable by codes of this type, which would be with the unshortened (73,45) code and full likelihood feedback.

We see from Figure 2 that we can obtain a bit error probability of 10^{-3} at an E_b/N_o of 2.1, or about 3.2 db. If we disregard the fact that errors are clustered in words, as perhaps we could do with scrambling, this would give a maximum improvement over no coding of about 3 1/2 db. Including the clustered effect might drop the improvement to less than 3 db, since we see that at 3.9 db (2.9 db improvement) the word error probability is 6.8×10^{-3}, about the desired 6-bit word deletion probability; this assumes practically all words with errors would be detected. On the other hand, if we confine ourselves to the shortened codes, then with the (49,21) it takes $E_b/N_o = 3.8$ db to get a 10^{-3} bit error probability. Detailed analysis of the distribution of errors in words yields the estimate that with this code and at this signal-to-noise ratio about 1% of all six-bit words contain errors; thus even with perfect detection of errors the word deletion probability would be slightly high. Incidentally, we see that with the (14,6) code, at the same signal-to-noise ratio, less than 2% of 6-bit words have errors; therefore this simpler code would probably be almost as good. But in summary, it would seem that in practice a gain of 3 db or more could be obtained only by skewing the transmitted bit rate with the unshortened code, and perhaps scrambling as well.

Next, let us compare the full likelihood with the hard decision feedback. We see from Table III that while the error probability in the information bits, P_{ei}, is only slightly less for full likelihood than for hard decision, the parity bit error P_{ep} is much better for full likelihood, being greater than P_{ei} for hard decision, but less than P_{ei} for full likelihood. The reason for this propagation effect with hard decision is as follows: in a block code with k information digits, the basic parity check

equation whose cyclic permutations give the orthogonal equations has a span of k+1 bits. For example, in the (73,45) code, the 46th bit is equal to a sum of nine preceding bits, the first of which is the first bit. Thus when we come to the parity bit in decoding, one of the equations will include only bits that have already been decoded, all of which in hard decision feedback will have infinite likelihood and will dominate all the other equations. The result is that the decoded parity bits are forced to be the bits in that code word which contains the decoded information bits. Therefore, if no information bit errors are made, no parity errors will be made. However, if only one information error is made, there must be at least d-1 parity errors, where d is the minimum distance of the code [5 for the (15,7), 10 for the 73,45)]. This multiplication of errors accounts for P_{ep} being much higher than P_{ei}. With full likelihood, on the other hand, it is not necessary to decode into a code word, and every decoded bit is more reliable than the one before. This suggests that while with hard decision feedback the information bits ought to be at the beginning of the word, with full likelihood they ought to be at the end. If this is done, there seems very little difference in performance between the two schemes, though full likelihood is a bit better.

4. Convolutional Codes with Sequential Decoding

Sequential decoding is an algorithm for conducting an efficient search for the most probable convolutional code sequence, given whatever is received. Since any available information can be programmed into it, it is highly flexible; in the current application, for example, it can be programmed not to consider any transmitted sequence not in the proper format, which includes both the seventh parity check bit and certain fixed words in every frame.

Because of the generality of sequential decoding, there are a wide variety of options in choosing a particular scheme. First, of course, is the choice of the code itself. For reasons discussed in earlier reports, we have limited ourselves to codes of rate 1/2. We have to date imposed a limit on constraint length of 24, both because we do not wish the encoder to become too complicated, and because the word size of the SDS computers available at the receiver sites is 24; we have now found that 25 can be handled by suitable programming. Also involved in the choice of code is the question of what to do with the seventh parity bit.

The decoding algorithm, besides having two variables called the bias and the threshold spacing, can be cast into two forms, the first due to Fano, the second to Gallager, which can appreciably affect implementation. In these simulations the Fano algorithm was used.

One must also choose a quantization scheme for the demodulator output; here for hardware simplicity we have limited ourselves to 3 bits or 8 levels, as the results of Appendix B suggest will be sufficient.

Finally one must specify how to resynchronize the decoder when it hangs up; here the simplest method seems to be to resynchronize once each frame by simply setting the encoder shift register to all zeros. Since this method leaves the last bits of a frame less protected than the rest, the best place to do this in the Pioneer format is after the second word of the frame, since the first word is the known frame synch Barker sequence, and the second the mode identification word, which is usually known but in any case dispensable.

Specification of the performance of a sequential decoding scheme is somewhat more involved than specifying that of the schemes previously considered, because of the variability in decoding computations inherent in the search algorithm. If a sequential decoder were allowed to search for an arbitrarily long time, it would eventually decode every frame, and then one

could speak of the word error probability as simply the fraction of decoded seven-bit words containing errors. In practice, a too-long search is terminated and the frame involved simply erased; one then speaks of the frame deletion probability. By varying the criterion of when a search is too long, one can decrease the undetected word error probability at the cost of increasing the frame deletion probability; as the choice of this criterion can be deferred until the data actually arrives, specification of the performance of a code must include the complete tradeoff which can be obtained.

A FORTRAN program has been written at Ames by L. Hofman to get estimates of performance with all these variables. We now discuss the results obtained.

In the matter of choosing codes, Lin and Lyne [5] have published codes optimized for sequential decoding of constraint lengths up to 21. Three different tails of length 3 were added to the length-21 code to get three different length-24 codes. Surprisingly, it was found that the differences in performance between these three codes were substantial; with a cutoff of 20,000 trials, the undetected error probabilities observed at E_b/N_o = 2.5 db and 1.7 db were: Code I, 3.6×10^{-4} and 1.1×10^{-3}, Code II, 1.8×10^{-4} and 4.9×10^{-4}, and Code III, 0 and 4.7×10^{-4}. Therefore, Code III was chosen; the tap configuration of this code is given in Appendix D.

The standard of performance was then taken to be this code with the following additional choices:

1) odd parity forced on all seven bits in a word;

2) fixed, known words in positions 15 ($\overline{\text{FS}}$), 31 (FS), and 32 (MID) of each frame;

3) a bias of 1.0 bit per branch;

4) a threshold spacing of 3.0 bits;

5) 3-bit quantization as in Figure 3a.

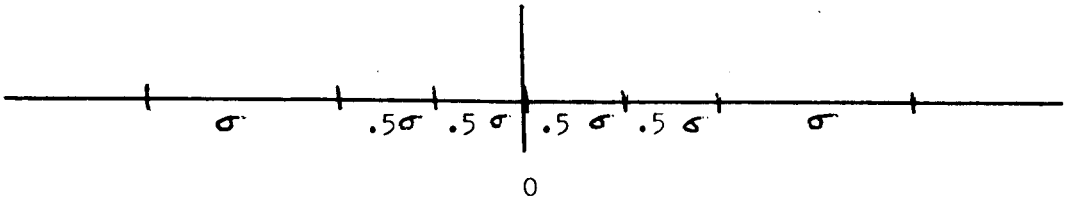


Fig. 3a. Unequal-step Quantization

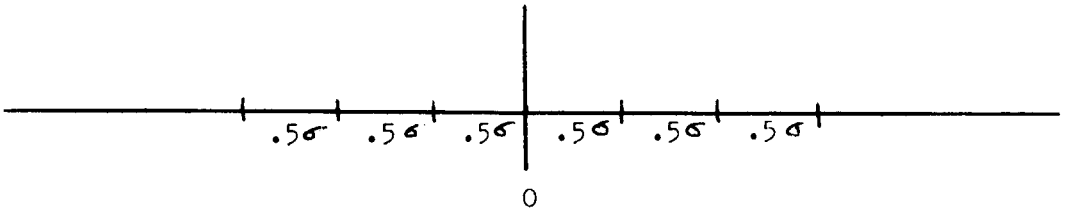


Fig. 3b. Equal-step Quantization

Note: Mean Output = $\pm E_b/N_o$

$\sigma^2 = E_b/N_o$

The performance obtained by this code, with these choices, and a cutoff of 20,000 trials, is summarized in Table IV, for E_b/N_o = 2.5, 1.7, and 1 db. Figure 4 illustrates the tradeoff between deletion and undetected error probabilities which can be obtained at E_b/N_o = 1.7 db by dropping the cutoff below 20,000 trials, while Figures 5 and 6 give the distribution of trials required in 4 runs of 800 frames at E_b/N_o = 1.7 db and 3 runs of 100 frames at E_b/N_o = 1 db.

Table IV

E_b/N_o (db)	N (frames)	P_{xl}, frame	P_e, word
2.5	3,200	9.4×10^{-4}	0
1.7	3,200	7.8×10^{-3}	4.7×10^{-4}
1.0	300	8.0×10^{-2}	2.4×10^{-3}

Use of the above code implies that the spacecraft encoder must substitute an overall parity check for the currently existing 1-3-5-7 parity check on all words that are not parity-exempt. As it is becoming clear that this function may represent a substantial part of the encoder (see Appendix D), a prime alternative to the above code would be to use the currently existing parity check. We have a limited set of simulations on this code, at E_b/N_o = 2.5, 2.1, 1.9, and 1.7 db, which are summarized in Table V, again with a cutoff of 20,000 trials. (In these simulations the equal-step quantization scheme of Figure 3b was used.) Figure 7 plots the tradeoff between deletion and error probabilities at the three higher signal-to-noise ratios, and Figure 8 plots the distributions of decoding computations.

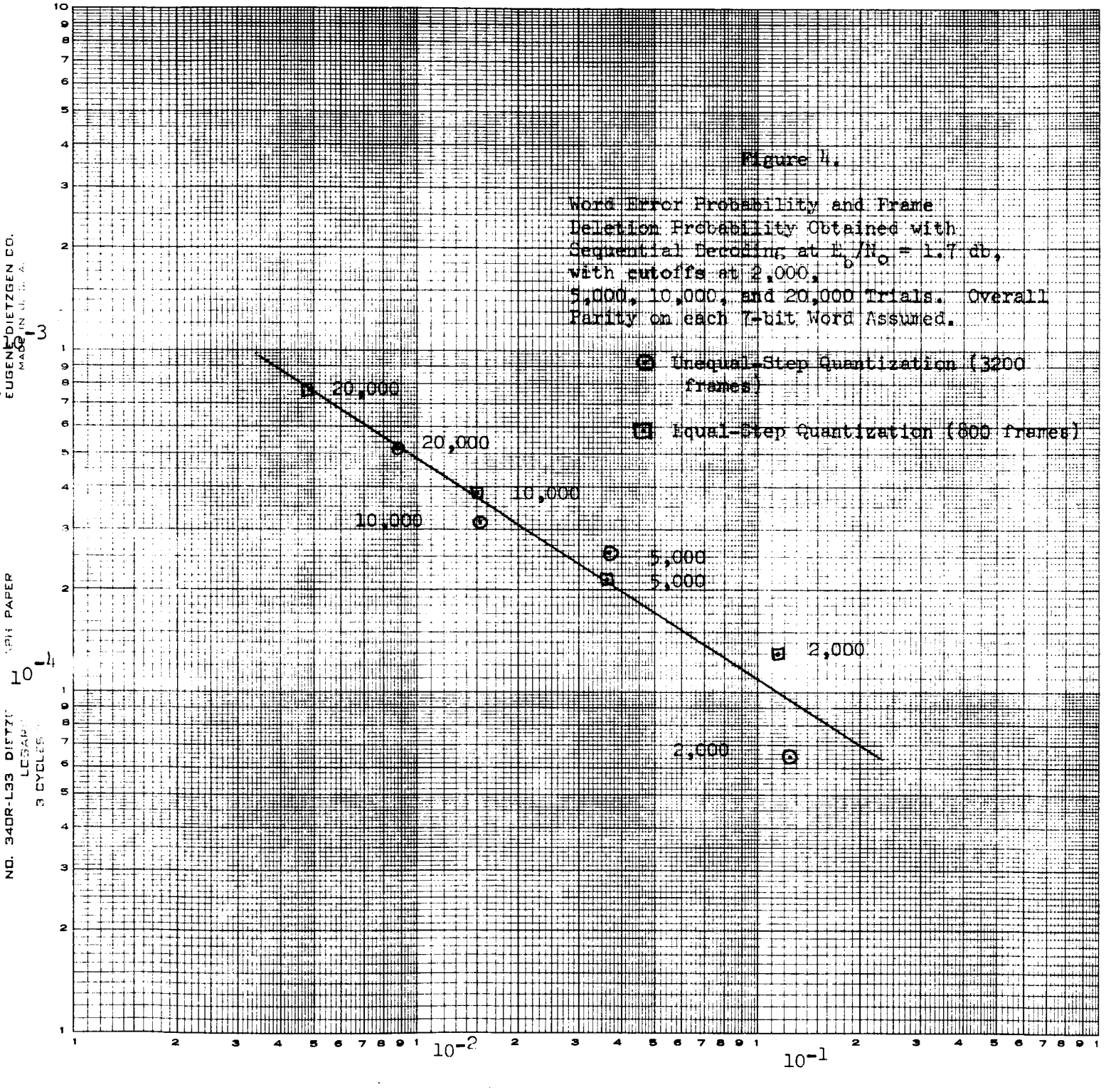


Figure 4.

Word Error Probability and Frame Deletion Probability Obtained with Sequential Decoding at E_b/N_o = 1.7 db, with cutoffs at 2,000, 5,000, 10,000, and 20,000 Trials. Overall Parity on each 7-bit Word Assumed.

- Unequal-Step Quantization (3200 frames)
- Equal-Step Quantization (600 frames)

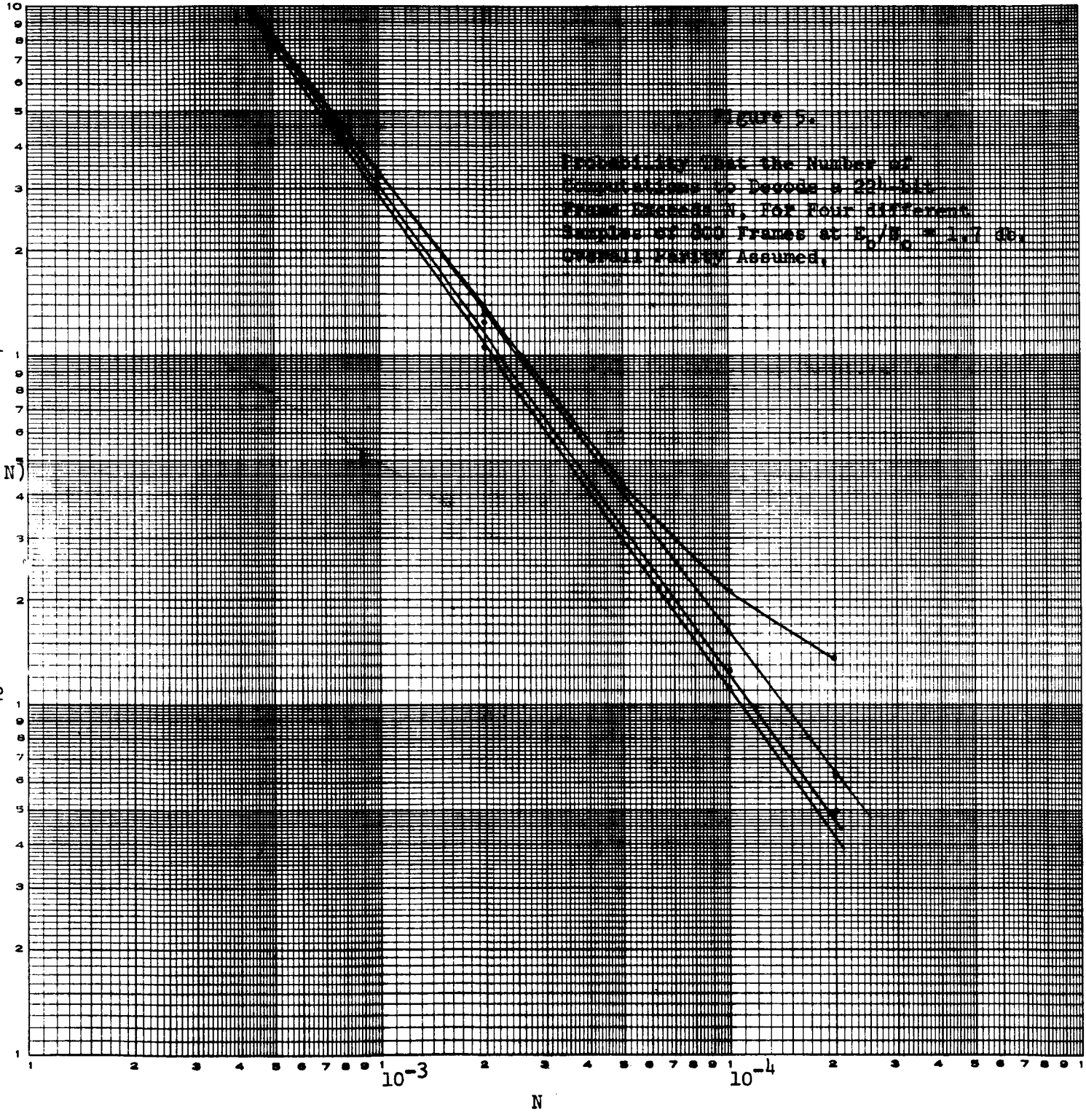


Figure 5.

Probability that the Number of Computations to Decode a 22[illegible]-bit Frame Exceeds N, for Four different Samples of 500 Frames at E_b/N_o = 1.7 dB, Overall Parity Assumed.

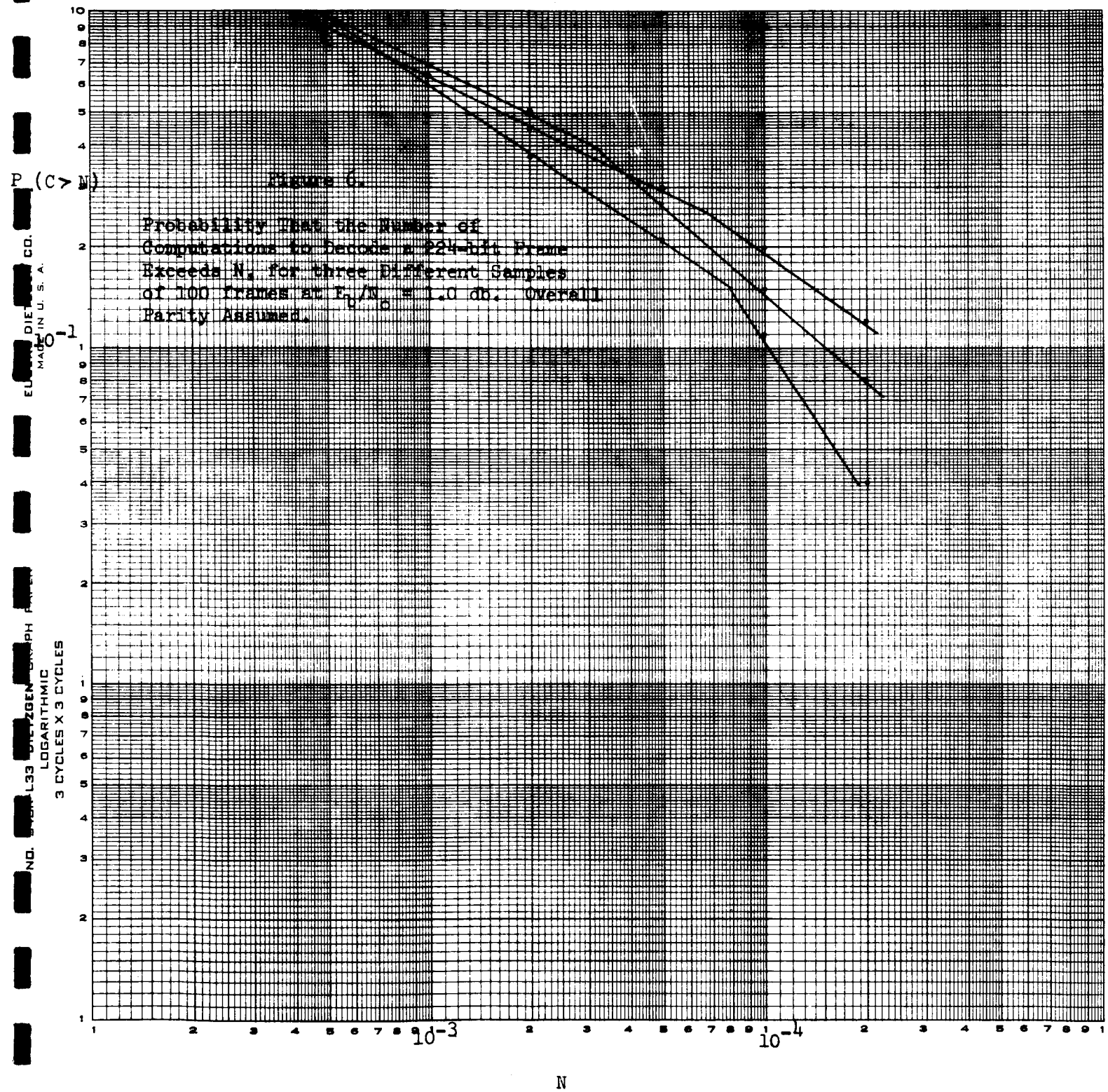


Figure 6.

Probability That the Number of Computations to Decode a 224-bit Frame Exceeds N_s for three Different Samples of 100 frames at E_b/N_o = 1.0 db. Overall Parity Assumed.

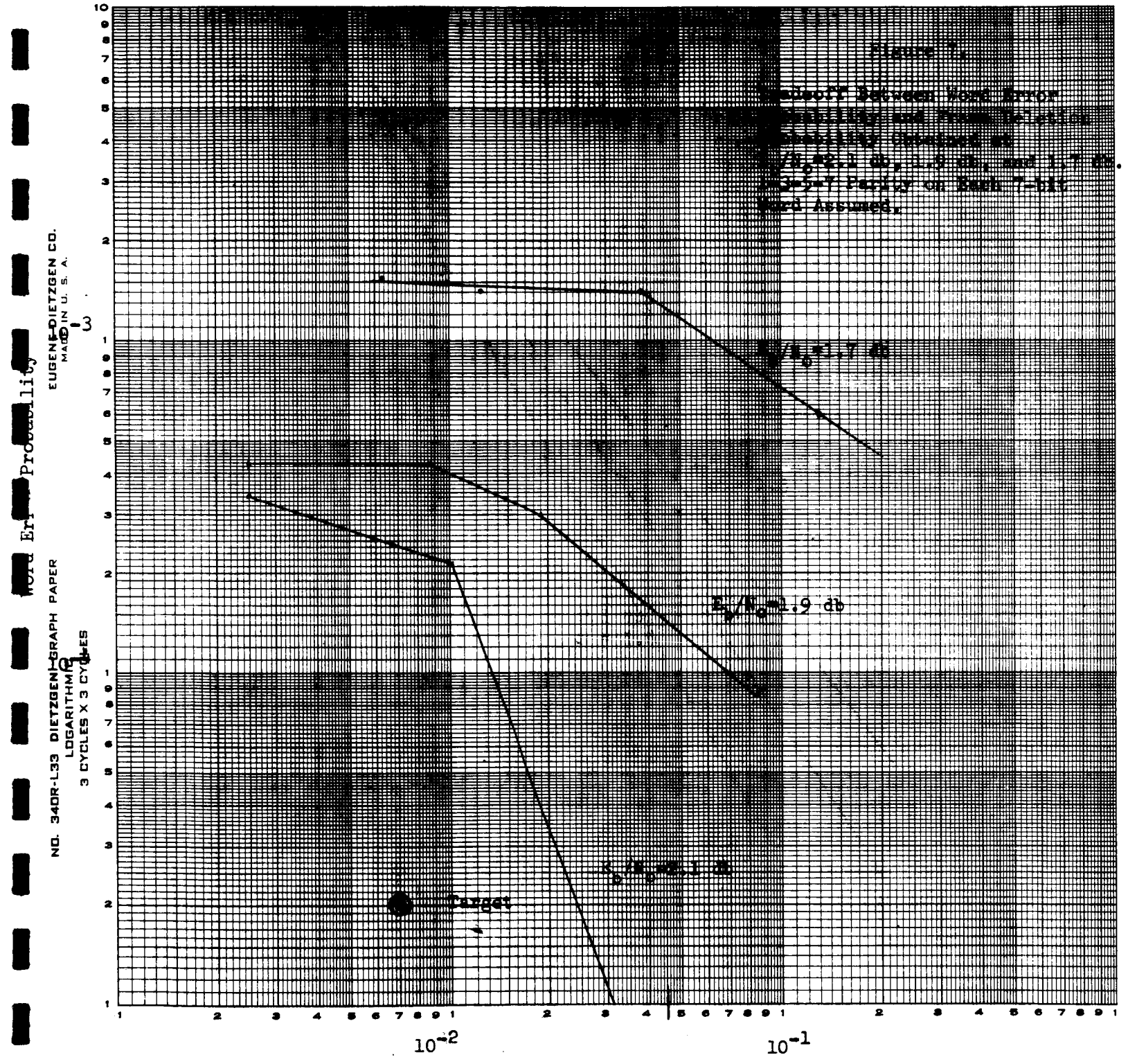


Figure 7. Tradeoff Between Word Error Probability and Frame Deletion Probability Obtained as E_b/N_o=2.3 db, 1.9 db, and 1.7 db. (8,7) Parity on Each 7-bit Word Assumed.

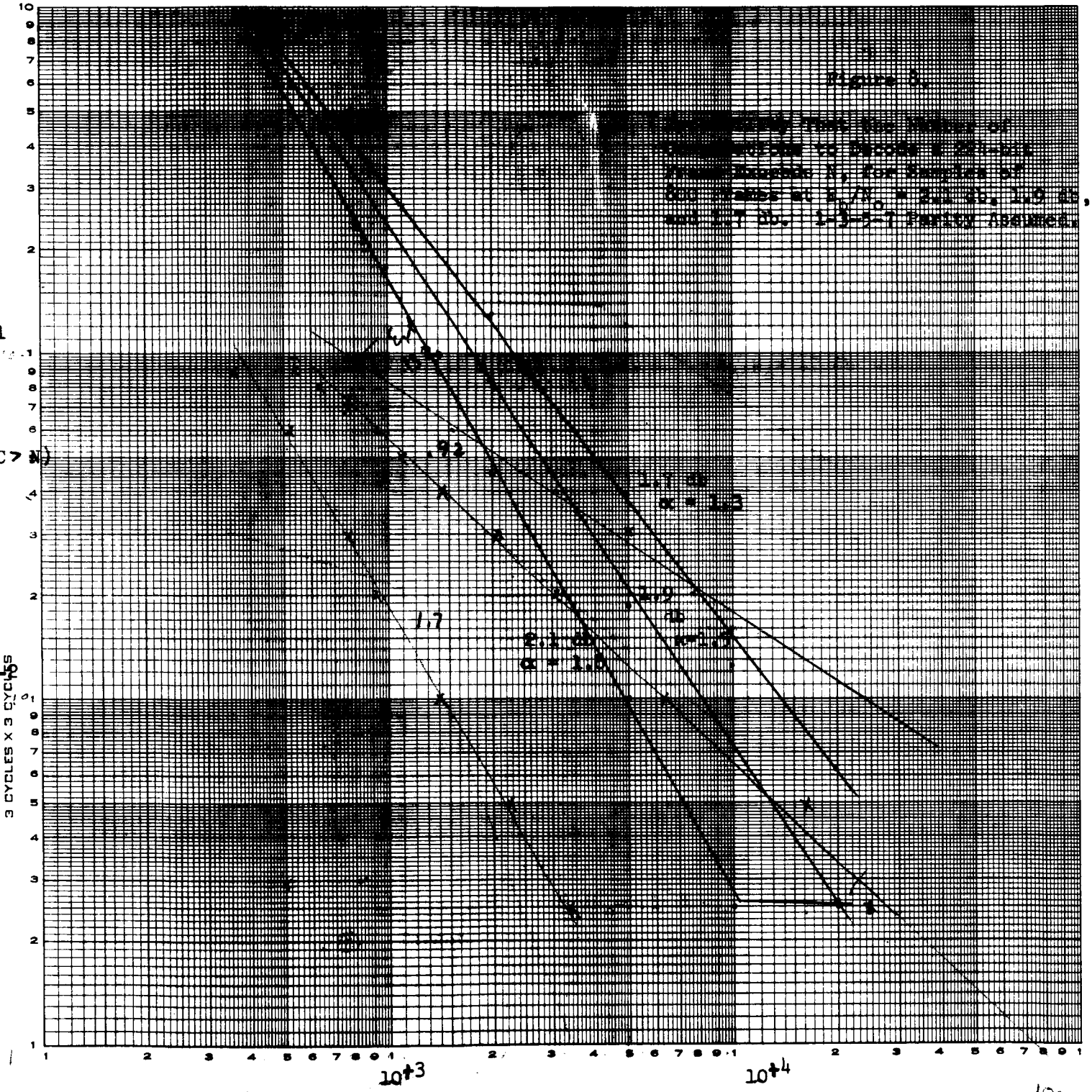


Figure 8.
Probability that the Number of Cycles to Become a 2-1b-inch Crack Exceeds N_f for Samples of 300 Taken at S_r/N_o = 2.1 lb, 1.9 lb, and 1.7 lb, 1-3-5-7 Parity Assumed.
$P_r(C > N)$
LOGARITHMIC
3 CYCLES X 3 CYCLES
N
10^{+3}
10^{+4}
.92
1.7
2.1 lb
$\alpha = 1.8$
1.9 lb
$\alpha = 1.3$
1.7 lb
$\alpha = 1.3$

Table V

E_b/N_o (db)	N (frames)	P_x, frame	P_e, word
2.5	800	1.3×10^{-3}	0
2.1	800	2.5×10^{-3}	3.1×10^{-4}
1.9	800	2.5×10^{-3}	3.9×10^{-4}
1.7	800	6.3×10^{-3}	1.4×10^{-3}

Finally, on the suggestion of Prof. Massey, the inverse of the (48,24) code was tried. This is a (46,23) code whose distance properties are identical, over the first 24 bits, to those of the (48,24) code. In one run of 800 frames at E_b/N_o = 1.7 db, an undetected error probability twice that of the (48,24) code was obtained.

Just after acquisition, or in the time neighborhood of a command, the mode identification word may not be known. Therefore, the performance of the standard code under the assumption that the last word in the frame was unknown, was determined. As expected, many errors were observed in the last word. However, frame deletion probability was unaffected and undetected word error probability increased by only a factor of 2 (at 1.7 db), most of which were in the last data words of the frame.

Finally, some preliminary runs were made with equal-step quantization of different sizes than in Figure 3b, but, aside from verifying that performance is not very sensitive to such changes, little can be concluded from the data obtained to date.

Figure 4 compares results at E_b/N_o = 1.7 db for the two quantization schemes of Figure 3.

DISCUSSION

The performance obtained with sequential decoding is quite impressive. At an E_b/N_o of 2.5 db, which is a 4.3 db improvement over no coding, we have yet to observe an error, while the deletion rate is still below one in a thousand. At 1.7 db, we are getting performance with the best code equal to the standard of deletion probability, and just a bit more than the standard of undetected error probability. We discuss below better methods of detecting errors; suffice it to say here, however, that with sequential decoding the improvement over no coding approaches 5 db. Perhaps more important, at about a 4 db improvement, almost absolutely clean data can be provided to the experimenters.

The most important comparison to be made between the different code configurations studied is between retaining the present 1-3-5-7 parity in each data word, or substituting an overall parity check in the seventh place. The fact that some words are parity-exempt makes such a replacement somewhat difficult to implement, as we see in Appendix D. From the fragmentary data we have, particularly from a comparison of Figure 4 with Figure 7, it appears that performance obtainable with 1-3-5-7 parity at E_b/N_o = 1.9 db is roughly equivalent to that of overall parity at 1.7 db, where in both cases those signal-to-noise ratios are close to the extremes that will be tolerated. The performance difference is thus of the order of .2 db. We question whether this small difference justifies the added complexity in the on-board encoder.

Figures 4 and 7 also tell us something about the sensitivity of total decoding trials as a deletion criterion. In both cases, in the range of interest, it takes about a fourfold increase in deletion probability to obtain a twofold decrease in error probability, whereas one would expect the error probability to decrease at least as fast as deletion probability increases. Furthermore, with 1-3-5-7 parity, the error probability becomes almost independent of deletion probability at large cutoffs. The moral is that the

total number of decoding trials is not a sensitive deletion criterion, particularly for the 1-3-5-7 parity. It appears that the number of trials needed to get past a short error cluster may be much less than 20,000.

This situation is particularly unfortunate in that, because of the rather short code constraint length, the ultimate performance of this scheme appears to be limited by the undetected error probability obtainable, not the deletion probability. A more sensitive deletion criterion, if successful in identifying the bulk of errors now undetected, might permit a gain of perhaps 1 db or so, particularly at the low data rates, before excessive computation would force a rate change or termination; Figure 6 shows that the computation cutoff comes near E_b/N_o = 1.0 db.

Therefore we consider it worthwhile to look for a more effective criterion. One possibility might be to count the number of trials required to advance a shorter length, say 24 bits or less; another would be to recompute the metric of the decoded sequence, and delete when too great a dip occurs. There is little urgency in such refinements, however, since the only implementational change they imply is in ground software.

The distribution of decoding computation illustrated in Figures 5, 6 and 8 is, as expected, of the Pareto form--that is,

$$\Pr (C > X) = KX^{-\alpha}.$$

Comparison of Figures 5 and 8, as well as all the other evidence at hand, indicates that this distribution is negligibly affected by what code is chosen, depending rather on the signal-to-noise ratio. The Pareto exponent can be estimated by curve-fitting as 1.8 at E_b/N_o = 2.1 db, 1.5 at 1.9 db, and 1.3 at 1.7 db. The sample size is really too small to tell much at 1.0 db, but it seems probable that here the exponent is at or below 1.0, which would account for the high variability in the results. It is this which leads us to believe that with extensive off-line processing, an E_b/N_o near 1.0 db could

be obtained if the undetected error probability problem could be solved.

Figure 4 indicates that the change in performance in going from unequal to equal-step quantization is negligible, as would be expected, and therefore we recommend the equal-step quantization for hardware simplicity. The exact size of the steps can be determined in further simulation, along with such other detail parameters as the bias and threshold spacing.

Finally, the simulations with the mode identification word unknown indicate that this situation is certainly tolerable if errors in the mode identification itself are tolerable. Incidentally, the flexibility of treatment of fixed words in the decoder suggests a possible method for extending range, assuming the carrier still trackable. If half or more of the experiments were shut down, a large part of every frame would contain fixed words; then the remaining data words would be decodable at signal-to-noise ratios below those otherwise required. Although the experimenter discipline required may be inconceivable, it might be worth seeing how much this effect amounts to.

REFERENCES

1. Phase I Report on a Study of Coding for Deep Space Telemetry, NAS 2-2874, Codex Corp., Watertown, Mass., October 8, 1965.

2. Phase II Report on a Study of Coding for Deep Space Telemetry, NAS 2-2874, Codex Corp., Watertown, Mass., March 28, 1966.

3. D. R. Lumb and F. Neuman, Error Rate Reduction of Parity Checked Telemetry Data by a Likelihood Deletion Strategy, NASA TN D-3576, August 1966.

4. J. L. Massey, Threshold Decoding, MIT Press, Cambridge, Mass., 1963.

5. S. Lin and H. Lyne, Convolutional Tree Codes for Sequential Decoding Systems, private communication.

Appendix B. Effects of Equal-Step Quantization

We have written a short computer program to calculate the capacity C and computational cutoff rate R_{comp} of the white gaussian channel with binary antipodal signalling, when the demodulator output is quantized in 2^m equal steps. Figure 1 illustrates such quantization for the case m=3; we have called the step size T. As in Appendix A, we assume the demodulator output is scaled to have mean $\pm 2\beta$ and variance 2β, where β is the signal-to-noise ratio per transmitted bit.

With quantization, the channel becomes a discrete memoryless channel with two inputs x_i and 2^m outputs y_j. The channel is completely characterized by the transition probabilities $P_{ji}(\beta,T)$ which give the probability of receiving y_j, given x_i, for particular choices of β and T. Information theory then tells us that the capacity of the channel in bits is:

$$C(\beta,T) = \frac{1}{2}\sum_j\sum_i P_{ji}(\beta,T)\log_2\frac{P_{ji}(\beta,T)}{\frac{1}{2}\sum_i P_{ji}(\beta,T)};$$

the theory of sequential decoders says that the computational cutoff rate in bits is:

$$R_{comp}(\beta,T) = 1 - \log_2\left[1 + \sum_j\sqrt{P_{j1}(\beta,T)P_{j2}(\beta,T)}\right].$$

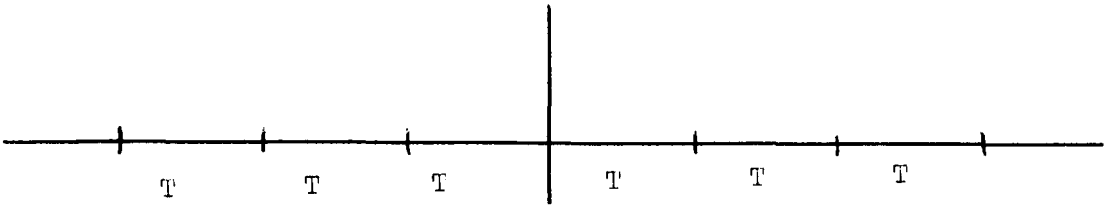


Fig. 1. 3-Bit Quantization

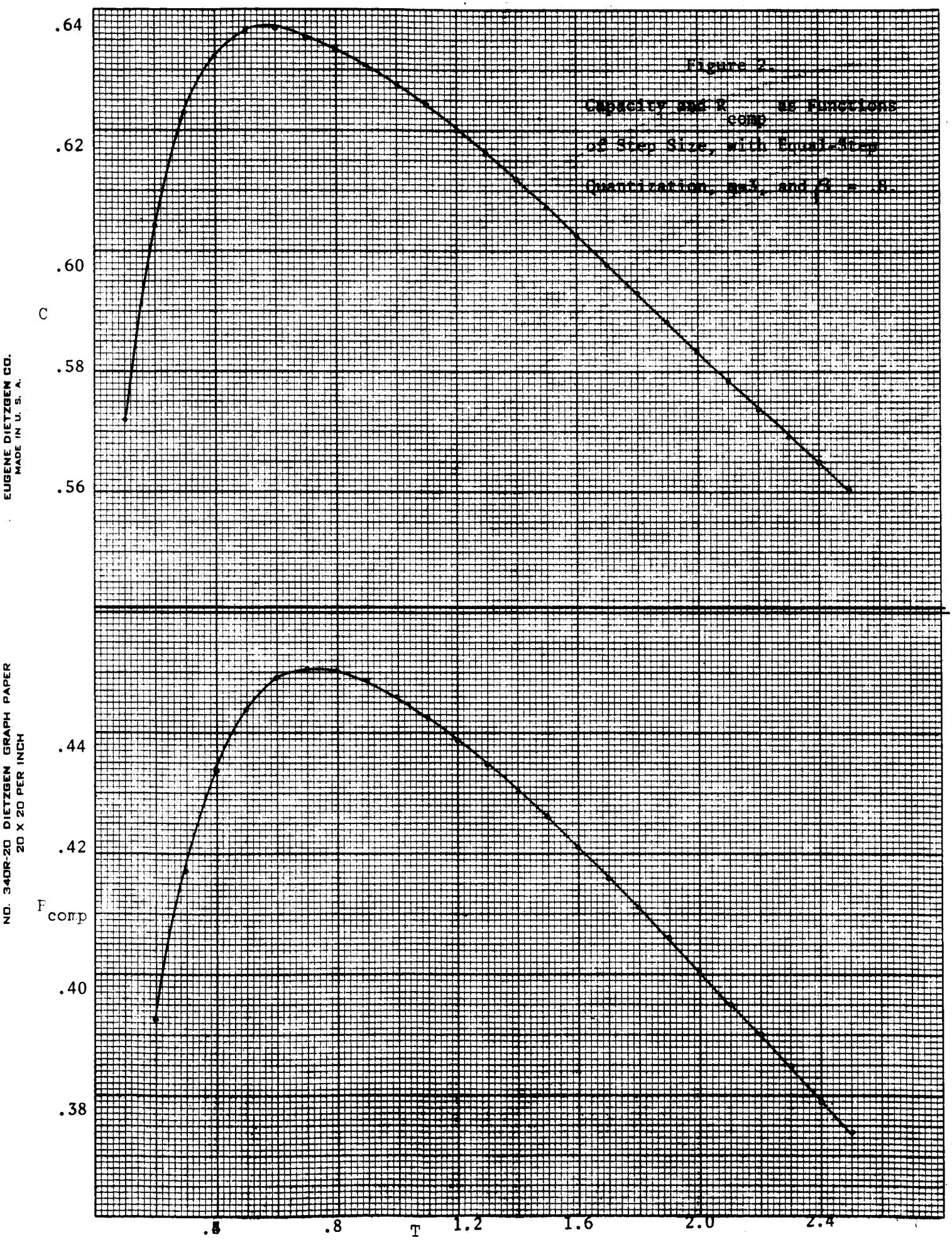


Figure 2.

Capacity and F_{comp} as functions of Step Size, with Equal-Step Quantization, $\mu = 3$, and $\beta = .8$.

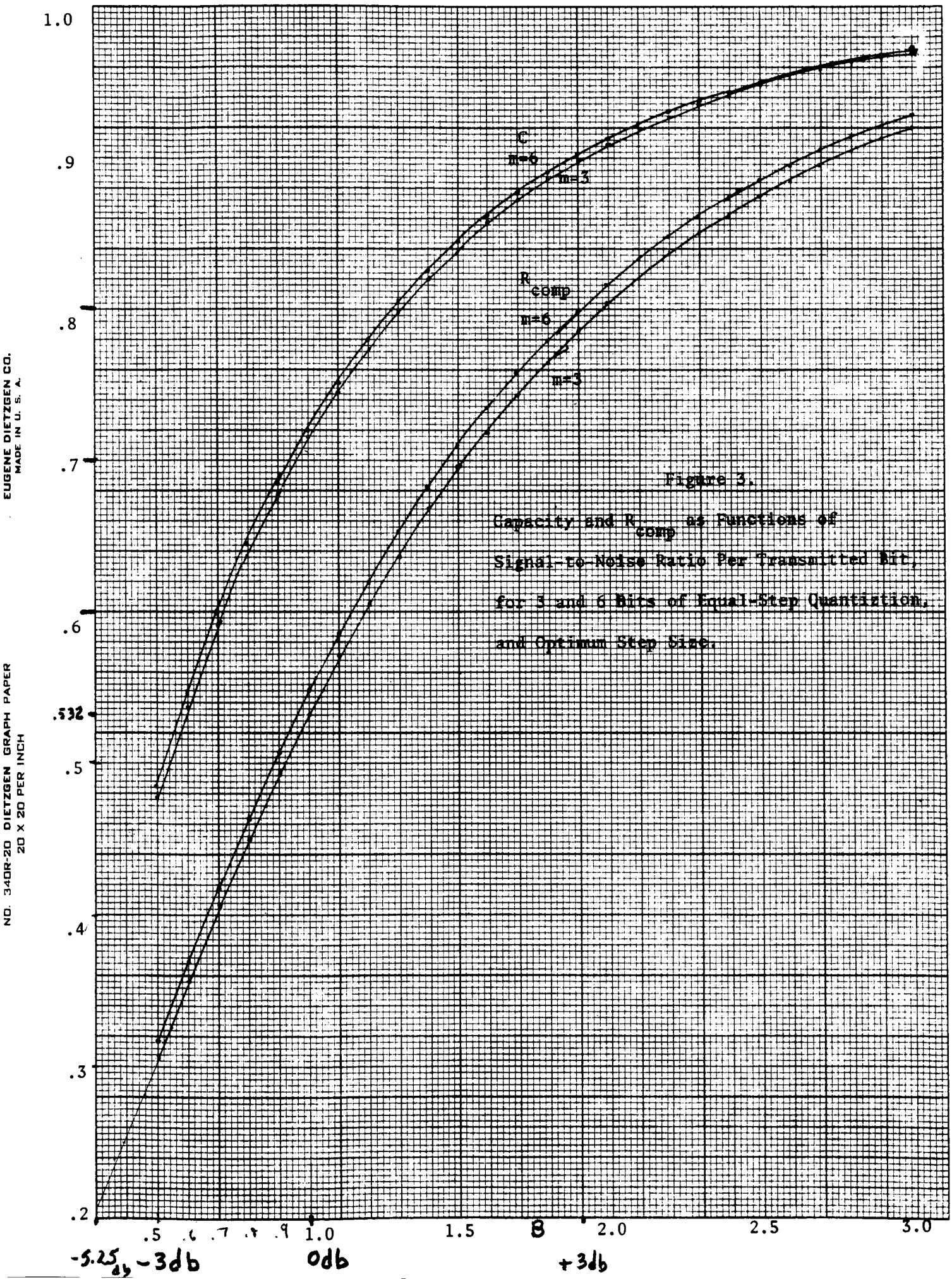


Figure 3.

Capacity and R_{comp} as Functions of Signal-to-Noise Ratio Per Transmitted Bit, for 3 and 6 Bits of Equal-Step Quantization, and Optimum Step Size.

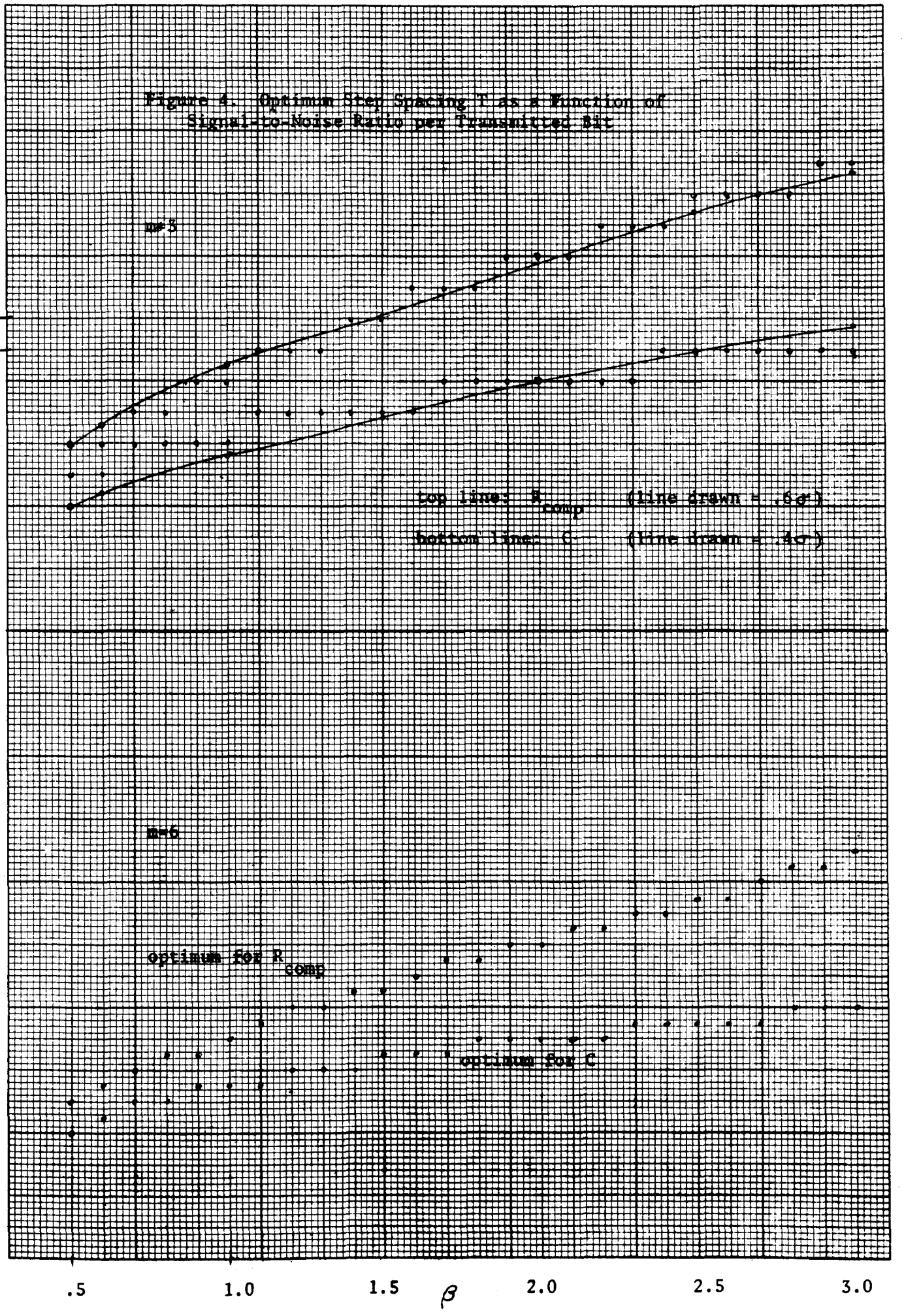


Figure 4. Optimum Step Spacing T as a Function of Signal-to-Noise Ratio per Transmitted Bit

Figure 2 plots C and R_{comp} as a function of T for m=3 and β =.8. We see that there is a definite optimum setting of T, and that this optimum is greater for R_{comp} than for C. These features were observed for all the curves we plotted, which included values of β from .5 to 3.0, and m=3 and 6. Figure 3 is a summary curve showing the C and R_{comp} obtained at these maxima versus β, for m=3 and 6. In the case of R_{comp} we have also included values for infinitely fine quantization, which cannot be distinguished from the m=6 curve. Figure 4 gives the optimum T as a function of β, for m=3 and 6.

Discussion

As the parameter which limits the performance of sequential decoding is R_{comp}, the most interesting graph is Figure 3. Here we see that the value of β required for R_{comp} = .5 with m=3 is less than .2 db above that required for m=6, which in turn is negligibly different from infinitely fine quantization. Three-bit quantization would therefore seem to be sufficient. Care must be taken, however, not to stray too far from the optimum value of T.

From Figure 4 we see that the optimum T is roughly proportional to $\sqrt{\beta}$ as was expected, or alternately to the standard deviation $\sigma = \sqrt{2\beta}$. In the case of capacity, $T_{opt} \simeq .4\sigma$, while for R_{comp}, $T_{opt} \simeq .6\sigma$. The latter figure checks with the fragmentary simulations alluded to in Appendix A.

Finally, we note the following rather puzzling fact. At E_b/N_o = 1.7 db, or β = .74, the simulations of Appendix A showed our rate - 1/2 code to be operating well below R_{comp}; rather than a Pareto exponent of 1, we observed an exponent of about 1.35. Yet, with m=3 and β = .74, R_{comp} is

only .42, with optimum equal-step quantization. Even taking into account the known seventh parity bit only reduces the effective rate to .43; we must also include the 3 known words in each frame to arrive at the lowest possible effective rate of .388, at which rate the computational cutoff comes at E_b/N_o = 1.2 db. It would seem that in our simulations we are observing the computational behavior which theory predicts for the best possible code of rate .388, even though our code is anything but optimum. Either we have discovered a very unlikely insensitivity in sequential decoders, or we must suspect that the random noise generator is slightly less noisy than was intended.

Appendix C. Machine Language Sequential Decoders

We have spent some time developing the fastest possible machine language sequential decoders for the SDS 910-920 and for the IBM 7094. The hope is to do the bulk of the decoding in real time in the SDS machines that are now or will be at the DSIF sites, and, as we see below, at the high bit rates (512 bps) the speed of decoding will determine at how low a signal-to-noise ratio we can work. For backup, to have another go at undecoded frames, and to process the outputs of listen-only stations, we must have an efficient decoding program for the 7094 at Ames. Finally, a fast 7094 decoder makes extensive simulations much more economical.

In developing these programs, we have valued speed much more than memory conservation. We have therefore written many almost identical parts of the program many times, rather than using subroutines, transfers or simple tests on flags, and have used table lookups wherever possible.

The 7094 program contains a random data generator, and a number of tabulating functions, the time and storage required for which we have not tabulated. It has an initialization section of 84 instructions which needs to be called only once for each run, or whenever the format is changed or the decoding parameters varied. The decoding routine occupies 2392 words of storage. At high signal-to-noise ratios, where the decoder proceeds rapidly forward through the tree, the average number of cycles per computation is 19 or 20, exclusive of tabulations; at low signal-to-noise ratios where searches predominate, this number drops to 15 or 16. Given

that the cycle time of the 7094 is 2 μs, and supposing an average number of computations per decoded bit of 10 or so, one could count on decoding about 10 frames a second as a long-term average, or 5 times real time at 512 bps, assuming no frames decoded earlier. For simulations, this would suggest a maximum decoding rate of 36,000 frames an hour, less whatever time is required for data generation and tabulation.

For the SDS machines, we have written two programs. The first attempts to be as fast as possible without regard for storage; the second tries to conserve storage without serious compromise to the efficiency of the program. Both include an interrupt routine which occupies 244 words, including 224 of I/O buffer storage. The fast routine also includes an initialization routine which consumes 32 x 224 cycles; thus, at 5 computations per decoded bit, 6 1/2 cycles or so must be added to the effective time of the fast routine. With this adjustment, the short routine takes only about 2 cycles more per computation than the fast routine; however, at 10 computations per decoded bit, this figure increases to about 5, and at a very large number of computations to 8 or more. In this last case the total number of cycles per bit is under 33 for the fast routine and under 41 for the short, a 20% difference. On the other hand, the storage needed by the short routine is only 1484 words including I/O buffer, as compared to 2705 for the fast routine.

With either routine we may estimate the average number of cycles per computation as about 40, when the number of computations per decoded bit is near 5. The interrupt routine requires about 20 cycles/received bit. Thus, the cycle time being 8 μs, at 512 bps nearly 6 computations per decoded bit (1200/frame) can be performed, if the machine time devoted to other tasks is negligible. At present about 25% of the machine time is being used at 512 bps; this would leave us 900 computations per frame.

The simulations of Appendix A indicate that this would yield a frame deletion rate of 2% and no undetected errors at E_b/N_o = 3.4 db, 9% at 2.5 db, and 36% at the ultimate cutoff of 1.7 db. Of course at lower bit rates these percentages will drop.

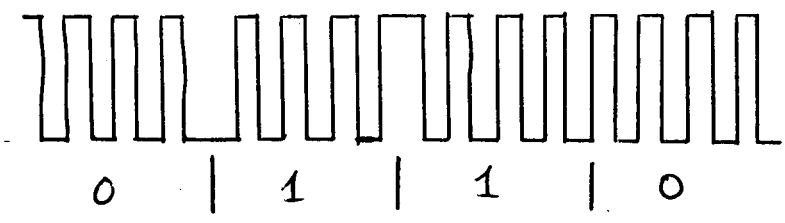
We conclude that an SDS 910 or SDS 920 could do a fairly decent job of decoding, if there were no other demands on it. Whether such a machine is usable or not depends on the time and memory requirements of these other demands.

Appendix D. Encoder Implementation

The following are some preliminary guidelines for implementation of the encoder suggested by the simulations of Appendix A. An initial hardware design based on these guidelines, using the TI series 51 integrated circuits, and omitting overall parity generation described below, indicates a requirement for 84 circuit packs.

1) Pioneer data format:

In the Pioneer spacecraft, the digital telemetry unit (DTU) has the function of collecting data from the experiments, formatting it and adding certain parity checks, and modulating the resulting data stream onto a square-wave subcarrier; its output is then fed to the RF modulator. Experimental data are in 6-bit words. The DTU adds a 7th bit to give odd parity on bits 1, 3, 5, and 7 ($b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 1 \bmod 2$). Data are transmitted in frames of 32 7-bit words. Of these words, all have the 1-3-5-7 odd parity except for three or four, depending on the format, which we call the parity-exempt words. After the addition of these parity bits and words, the bit rate is 8, 16, 64, 256, or 512 bits per second. These data are finally differentially modulated onto a 2048 cps square wave; that is, a data 'one' causes a 180° phase shift from the previous phase; a data 'zero' none. Thus a 0110 input at 512 bps would cause the following output:



2) Encoder functions:

The first thing the encoder must do is to reconstruct the data. This could be done, for example, by sampling and holding at T-second intervals, where T is the inverse of the bit rate, to take out the subcarrier; then a mod 2 adder and 1-unit delay could take out the differential modulation, as below:

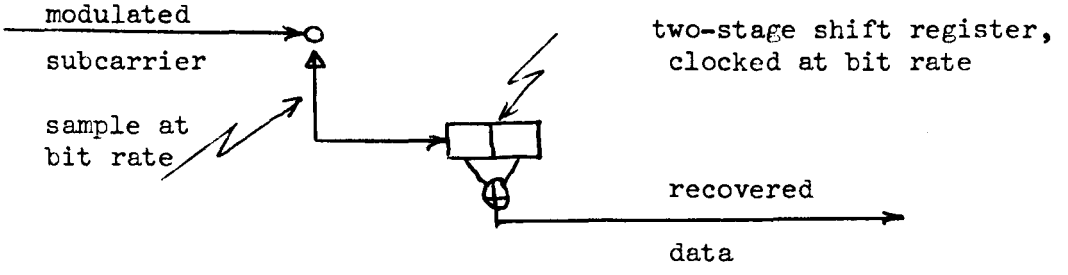


Fig. 1. Reconstructing the data

It is then desired to remove the (1-3-5-7) parity bit generated by the DTU, and to replace it with a parity bit giving the whole word odd parity. This can be done, however, only on the words which are not parity-exempt. Thus we need as inputs not only the word rate pulses, which occur on the seventh bit of each word (with skewed timing), but also either word gates for the parity-exempt words, or internal timing and gates to do the same thing. Figure 2 illustrates how this might be done.

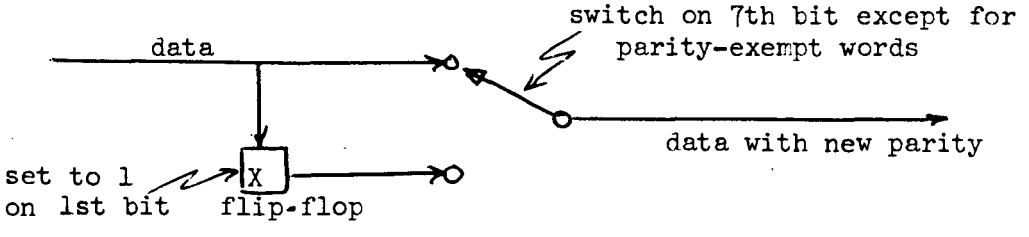


Fig. 2. Replacing the parity bit

It will probably be complicated to develop the timing for this parity regeneration; since the improvement gained in performance is small, we would also like a design in which this function is omitted.

Next, we encode the data in a rate-1/2 encoder of length 24, as shown below. The taps are to be as indicated.

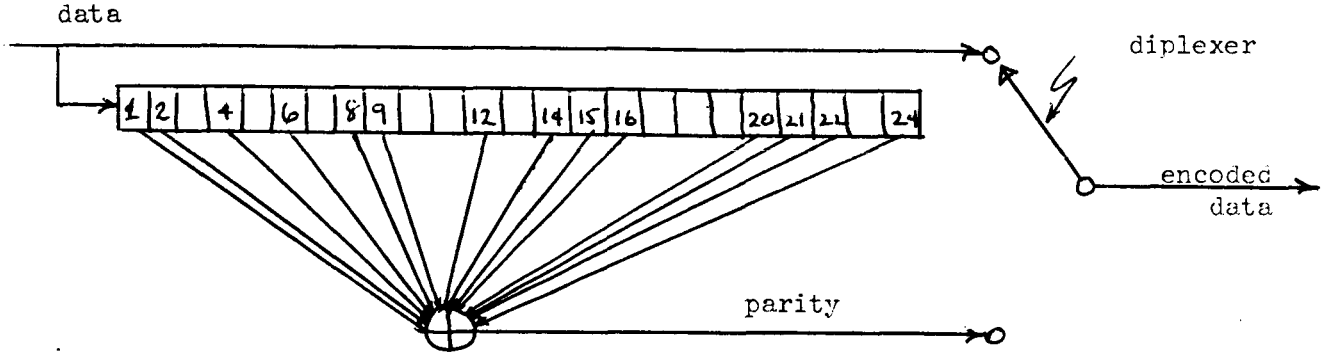
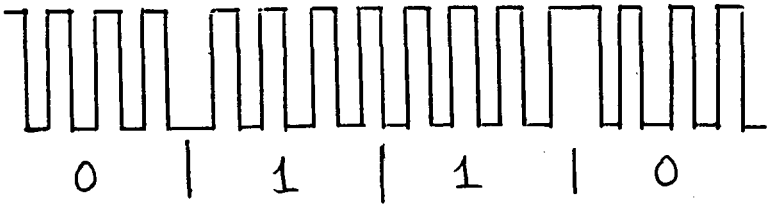


Fig. 3. Encoder

For resynchronization, it is desired that after the seventh bit of the 2nd word of the frame enters the encoder and the corresponding parity bit is computed, the contents of the encoder shift register be set to all zeroes. The timing for this could be developed from the frame rate pulse, or could be got directly from the word 3 word gate.

Finally, we want to remodulate the encoded data onto a square-wave subcarrier. At 256 and 512 bps, the frequency of this subcarrier is to be 2048 cps, as before; at the lower rates (64, 16, and 8 bps), it is to be 512 cps. Modulation is direct bi-phase PSK, not differential; thus a 0110 input at 512 bps (transmitted; 256 information bps) would cause the following output:



Clearly this modulator is no more complicated than a mod 2 adder, as below:

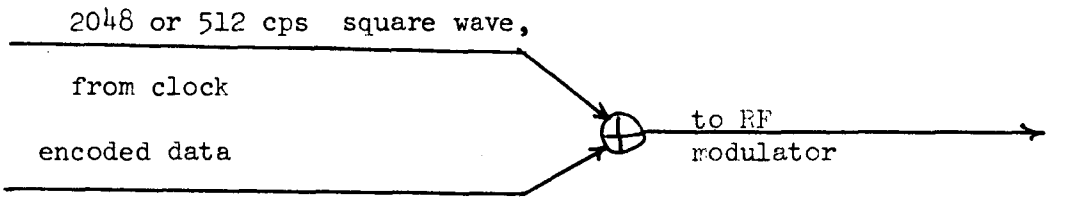


Fig. 4. Modulation